University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Translating Sign Language using Computer Vision and Methods of Machine Learning

William Traynor

March, 2022

Abstract

There do not exist any accurate sign language translation models, this issue is primarily due to the data available.

This project built an end-to-end sign language translation system for British Sign Language into English with a method of data collection that solved the main privacy concerns of prospective volunteers. This could increase likelihood of participation and be a possible avenue to conduct sign language data collection on a larger scale.

Education Use Consent

Contents

1 Introduction

Sign language has been a documented part of Western culture since the 1600s as a medium for communication (wik 2022). Sign language is a series of gestures made with one or both hands, sometimes helped with facial expressions, mouthing of the words and finger-spelling for uncommon words or proper nouns. Sign language also boasts a grammatical form different to Western languages, for example, to sign the phrase "It's time to sleep" a person would perform the sign for "time" followed by the sign for "sleep". Furthermore, there is no universal sign language, there are sign language equivalents for many world languages like Japanese, Spanish and English. Furthermore, sign language can differ geographically. For example, there exist many English sign languages like American Sign Language (ASL), Australian Sign Language (AUSLAN) and British Sign Language (BSL). The translation for this project will be from BSL to English. All of the aforementioned features of sign language make for a difficult translation problem. It is also a visual language so poses the further problem of having to learn what hand movements convey specific words and how multiple signs are combined to make phrases.

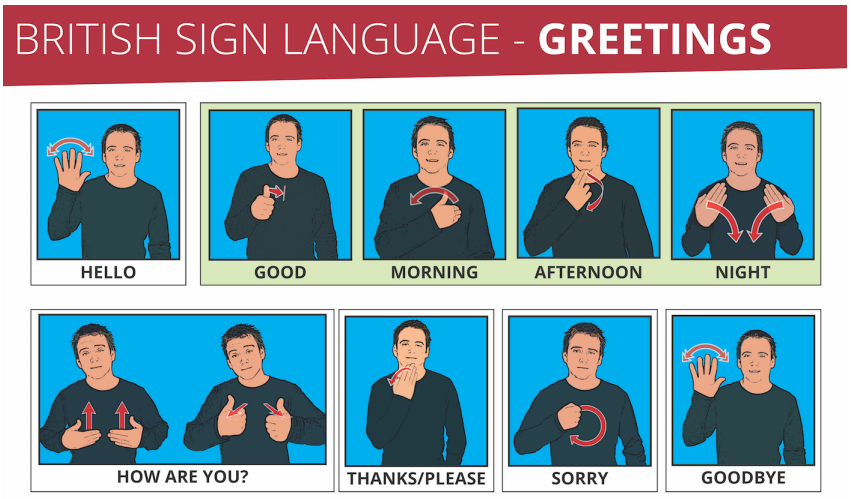


Figure 1.1: *Examples of common greetings in British Sign Language obtained from https://www.british-sign.co.uk/bsl-greetings-signs-british-sign-language/*

1.1 Motivation

There are 11 million deaf or hard of hearing people in the UK and only around 150 thousand people able to speak sign language (bri). This shows there is a large number of people for whom sign language may be their only form of communication yet only a small proportion of the population know it. Improvements in sign language translation could see a solution to helping those who are deaf or hard of hearing feel less alienated from the general population.

In recent years there have been major advancements in the field of natural language processing (NLP) with the likes of BERT, eLMO and GPT-3 whose contribution has allowed for very accurate translations between written languages. However, the work in sign language translation has had difficulty replicating the improvements seen in NLP. One reason for this is that, in the past, the focus was put on the recognition of individual signs being made rather than the translation of a sequence of signs into a written language (Wang

et al. 2016) (Camgöz et al. 2016). Unfortunately, the individual translation of signs with great accuracy has little effectiveness in the real world. Vahdani et al. (2022)

With sign language translation there is also a shortage of large, well-labelled data sets. The lack of useful data makes the already difficult problem that much harder.

This research project will focus on translating video footage of BSL performances, focusing only on the movement of the signer's hands, disregarding other features like face and arm movement. The translation will consist of a computer vision solution used to obtain hand pose data from the video footage as well as methods of machine learning to then develop a model which will predict an English translation from a video sign language performance.

1.2 Aims

This project aims to;

- Implement an end-to-end solution for translating BSL video footage
- Create a new data set to be used for training a BSL-English translation model

1.3 Chapter Outline

Here I define the structure of the project. Important chapters are the design and implementation as they explain the creation of the data set as well as the augmentation methods.

- **Chapter 2 - Background**: The prerequisite reading and research I carried out to understand how best to approach the problem.
- **Chapter 3 - Requirements & Use Cases**: The requirements and use cases for the project.
- **Chapter 4 - Design**: A description of how each stage of the translation system was created to allow for end-to-end translation.
- **Chapter 5 - Implementation**: A thorough explanation of how each section of the system was executed. This section further delves into the data set used and how the model functions.
- **Chapter 6 - Evaluation**: Evaluation of the final model and comparison with models built with differing configurations in the hope to improve performance.
- **Chapter 7 - Conclusion**: A summary of how the evaluation compared with the initial aims for the project as well as definitions for future work I would've carried out given a greater time allowance for the project.

2 Background

2.1 Deaf Culture

People who are born with the inability to hear would identify themselves as Deaf. The capital "D" spelling refers to the cultural label, and the lowercase spelling refers to the medical condition. The Deaf community brings together those with shared experiences through a common form of communication that is sign language. The Deaf community cannot be defined simply in medical terms, it is a group that is more complex and nuanced. The quote here from Baker and Padden (1978) best describes the Deaf community.

> "The Deaf community comprises those deaf and hard-of-hearing individuals who share a common language, common experiences and values and a common way of interacting with each other and with hearing people. The most basic factor determining who is a member of the deaf community seems to be what is called 'attitudinal deafness'. This occurs when a person identifies him/herself as a member of the deaf community and other members accept that person as part of the community."

The minority of people who are deaf are born deaf with most losing their hearing as they grow older. Usually, these people are not BSL users, with most BSL users being born deaf. Deaf people are normalised to deafness and consider their situation as a different human experience, whereas those who lose their hearing consider deafness as a disability. The Deaf community use BSL as their first language whereas people who have lost their hearing will still use English (bsl).

2.2 BSL Grammar

The grammatical format of BSL is different to that of English. English follows a "subject verb object" structure and BSL follows a "topic-comment" structure. In this structure, topics are what is being spoken about and the comment is what is being said about the topic. The difference is easier to see in an example.

English : "I made some pancakes"

BSL : "Pancakes I made"

The grammar of BSL is an aspect of this problem that makes it difficult to perform end-to-end translation. This is because to make a translation we have to translate the phrase in the correct order it would be read in English but we also need to predict the filler words which then make the sentence readable. However, this does not make the problem impossible as there are instances of successful translations for other topic-comment languages like Korean and Hebrew. (Aiken 2019).

Another aspect of BSL that can make it difficult to translate is that you have regional signs that will only be used by a given geographical subset of the country. For instance, there exist signs used in the north of Scotland that would not be understood in the south of England (Linguisttraining.com 2016). It is then difficult to obtain sufficient data to be able to translate these regional signs. This issue is comparable to the problem of accents in speech recognition.

2.3 Related Work

In sign language translation (SLT), one of the main issues that arise is the scarcity of well-labelled data.

The SLT problem can be viewed as an extension to sign language recognition (SLR). Here we will make the definitions, similar to the distinction made in the Camgoz et al. (2018) paper;

1. *Sign Language Recognition*: Mapping a single sign language gesture to a single word.
2. *Sign Language Translation*: Taking a sequence of signs and returning the meaning of the entire sequence in written language.

2.3.1 Sign Language Recognition

SLR work dates back to the late 80s and most of the initial work was completed through the use of motion sensors for hands and body movement, with Tamura and Kawasaki (1988) being the first to tackle the SLR problem for Japanese sign language. These sensors lead to fairly accurate solutions, however, due to the cost of the motion devices, saw no real-world applicability.

The work done by Vahdani et al. (2022) saw a 92.88% accuracy when recognising 100 American Sign Language (ASL) words. This work saw the creation of a data set for 100 ASL signs from 15 different signers using the Kinect v2 camera. This is a positive result for this project as it shows we are not trying to run before we can walk.

SLT is an extension of SLR so the fact that hand gestures can be translated very well into written language allows the conclusion that inaccurate SLT is due to an aspect of the problem that extends SLR. I should note this work made translations considering hand poses and facial expressions, for this project I will solely consider hand poses. Vahdani et al. (2022) had to create a new ASL data set, demonstrating one of the introductory points that data scarcity makes the already difficult problem even more taxing and expensive.

2.3.2 Sign Language Translation

Camgoz et al. (2018) attempted to make translations of sign language phrases into written language. This work had to tackle the grammatical details of sign language when predicting a translation. They used the RWTH-PHOENIX-Weather database which consisted of 7 thousand sentences with a 1081 sign vocabulary from 9 different signers. The translations for this data set are from German Sign Language to German. The paper by Camgoz et al. (2018) also practices viewing the SLT problem similar to how NLP solutions are constructed through using neural machine translation defined by Bahdanau et al. (2015) which tried to tackle the issue seen with standard encoder-decoder models in that translation accuracy deteriorates as the length of input increases. This architecture seemed like the natural choice for this project since the nature of the input for sign language is going to be longer than the predicted output. Camgoz et al. (2018) proposed the first end-to-end solution for SLT.

2.4 Data

While data scarcity is an issue, the catalogue of sign language data sets is not empty. There are data sets available that use sign language interpreter footage from news and weather broadcasts and pair them with subtitles from the broadcast. This amounts to labelled sign language data. These data sets are diverse in vocabulary and easier to collect as they don't require any filming which can be a very time consuming and costly task. However, a limitation to these data sets is that they are still not large enough to solve the SLT problem for the broad vocabulary they encompass. Another limitation is that the data sets are not always accurately labelled as they rely on the subtitles synchronising with the sign language interpreter. Finally, the video data comes from a small number of signers meaning there is not large diversity in how the signs are being performed. This project posed an alternative data set in the hope it would not be bound by the same limitations.

In this paper, I will create a data set with a greater number of signers to the number of phrases ratio which will increase the variance in performance for a given phrase. I feel this should improve the model and allow the model to be accurate for a greater number of people. This is because, like spoken language, sign language can be done differently by different people, some people have different hand sizes and may sign at different speeds. However, the task of obtaining sign language videos from many different volunteers can be very difficult, timely and expensive.

There are also privacy concerns that can make volunteers reluctant to participate in creating videos for a potential data set. The Bragg et al. (2020) paper attempted to address privacy concerns while maintaining

algorithmic performance on machine learning tasks. They proposed distortions of training sets that could increase the privacy of the signer. They saw that the main concerns shared by deaf and hard of hearing as well as hearing people were video misuse (more so for public data sets) and revealing sensitive information like surroundings, being recognised by people and signing personal content. This paper concluded that "privacy concerns may be pervasive in the community, and that filters may impact willingness to participate". The method of data collection proposed in this project looks to solve the main privacy concerns because of how the data is collected and what data is used for training.

2.5 MediaPipe

With any sign language problem, the first hurdle is converting video data to a useful form that can be used to train a model. This requires a computer vision solution. Data sets used in previous work have been cultivated through analysis of sign language videos and training of models which can pick out the hand pose. Some work also used facial expression and mouth movement analysis, for example, the Vadhani et al. work previously mentioned used facial analysis. Also, the Phoenix data set used by CamGoz et al. had facial expression data for each frame.

Figure 2.1: *Demonstration of how the PHOENIX data set annotated a signers facial expression Bragg et al. (2020).*

The task of obtaining hand pose from our sign language videos for this project was left to the *MediaPipe Hands* (MPH) solution. This solution was used to recognise the 3D pose of hands in an image. In this project, we fed each frame of the video into the model to retrieve hand pose data for each frame in a video. This allowed us to have an accurate hand pose data set for each video collected.

Figure 2.2: *Demonstration of how the MediaPipe solution picks out the pose of a hand in a given frame.*

2.6 Sequence to Sequence Models

The model we use to make the translations is an extension of a sequence to sequence (seq2seq) model. These models were first introduced by Google in 2014 Sutskever et al. (2014). The aim for seq2seq models is to map an input of fixed length to an output of fixed length, where the input and output lengths can be different. These models are a special type of recurrent neural network (RNN). The use cases of seq2seq models are generally NLP problems like language translation, text summarizing and creating chatbots. A seq2seq model

extends an RNN because rather than use a single RNN for a task like translation a seq2seq model will use 2 RNNs. A single RNN can take a sequence as input and produce a sequence as output, however, the outputs are produced in real-time meaning for each word in a phrase inputted, the RNN produces a word as an output before the next inputted word is given. This means that a translation would not be able to see any future words. This idea is depicted in figure 2.3.

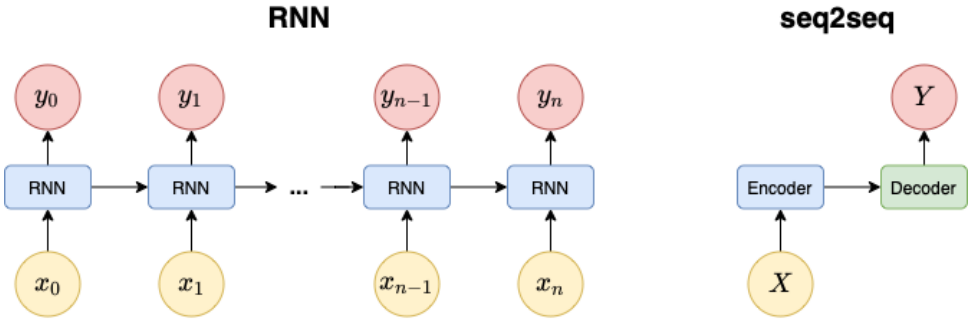


Figure 2.3: *Comparison of basic RNN and seq2seq diagrams to demonstrate how the outputs are produced in regards to the inputs*

In contrast, a seq2seq model would wait until it has received the entire input sequence, X, before producing the output sequence. This requires two RNNs commonly known as the encoder and decoder. A diagram of this is shown in figure 2.3.

3 Requirements & Use Cases

3.1 Requirements

The goal of the problem was to have a system that could take video footage of a sign language performance and output the sign language phrase being performed in English. My approach in trying to solve this problem was to improve the data used. I attempted this by increasing the diversity of signers in the data set. I wanted to train the model on data that was concise to ease the task of determining what a sign means and focus more on learning the complex grammar of sign language.

3.1.1 Participants

We attempted to gain video footage from a lot of different participants to create a more diverse data set. Using data sets from a small number of signers could be an area that will hinder the model when it comes to real-world testing of a new person performing sign language. We think a reason that NLP has become so efficient is due to the massive corpus' of data that can be used to train models. A key feature, we believe, of these data sets is that the entries have come from many different people, with different writing styles and vocabularies. Furthermore, we think this area for sign language would be even easier to compare to voice recognition. In voice recognition, it would be uncommon to have two people who sound exactly alike. So in the task of identifying what word is being said the model has to try and blind itself to accents and idiosyncrasies in speech and learn the principal phonetics that can identify a given word. This was the idea behind increasing the ratio of participants to phrases. we wanted the model to be able to find the defining gestures that make up a phrase amongst the uncommon mannerisms a given signer may have had during their performance of the phrases.

3.1.2 Data Size

Due to the constrained time and resources, we had to try and minimize this problem while still having a product that could show proof of concept. So, while we worked on a small phrase set of 22 phrases. We chose a set of phrases that had overlapping words so that our model could see the same words in different concepts. Furthermore, to educate the model on the unusual grammar sign language possesses we chose phrases where the same sign could represent different sub-phrases. For example, take the phrases "are you worried" and "what is your name". When performing these phrases a signer will do the same gesture for the "are you" part in the first phrase and the "your" part in the second phrase. We created the data set trying to include as many occurrences of overlapping words and identical gestures for different sub-phrases while maintaining a relatively diverse vocabulary.

3.1.3 Minimal Input Data

Our interpretation of the problem after reading the related work was that the difficulty in translation arises due to a large amount of input data to the model for a single translation. We are attempting to translate 100-150 frames of video into phrases with 4 or 5 words. So, we think in order the gain the most effective model it could be helpful to reduce the number of features in our input data for each frame to reduce the complexity of the data.

3.1.4 Fine Tuning

As with any machine learning problem, we should also ensure that we choose a set of parameters that impact the model performance. From there we want to tune these parameters so that we see the best performance

from the model.

3.2 Use Cases

The proposed solution for this project would be an ML model that can predict an accurate translation for a given sign language performance. A solution like this would be helpful for the Deaf community. It could help connect those who can and cannot use sign language. The use of an accurate SLT model can be compared to that of a good speech recognition model. It could allow the implementation of;

- On-screen captions for a signer when on video calls.
- Quick translation applications to be used by someone who cannot speak sign language.
- Teaching tools to give feedback to someone trying to learn sign language.

We also think the creation of an accurate SLT model could help raise awareness for the deaf and hard of hearing community. This could result in even more effective solutions like having mandatory sign language lessons for school children.

4 Design

To solve the problem of end-to-end translation of BSL the system set-up requires a method for obtaining useful data from video input and using this data to train a machine learning model to make

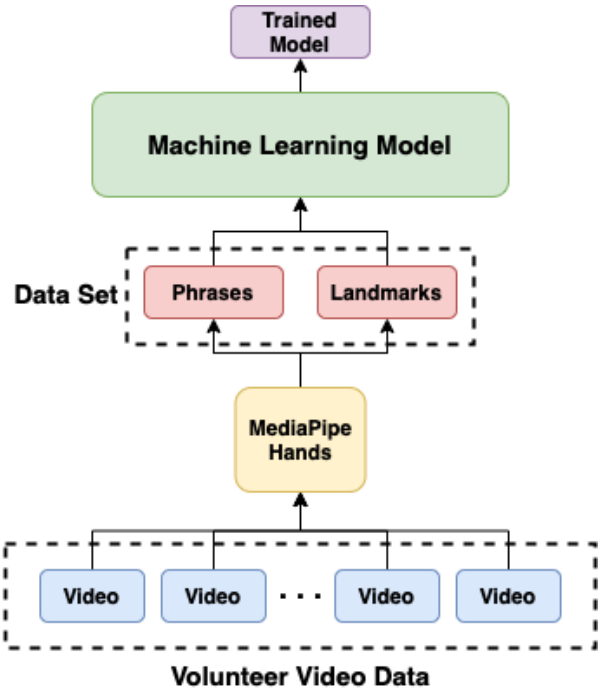


Figure 4.1: *Basic system diagram of the entire project from videos to final trained model.*

4.1 Data

An important aspect of this project is that the data set was created from scratch.

4.1.1 Video Collection

Before we looked for volunteers to record themselves performing BSL we first had to define a set of BSL phrases that would make up the data set. We also had to find a way to show the volunteers how to sign the phrase, otherwise, our possible volunteer set would be cut down to people who already knew sign language. With that, we found a collection of "learn to sign" videos on the Guy's and St Thomas' NHS Foundation Trust's YouTube channel (Lea). The collection consisted of 50 videos that demonstrated how to perform a given sign language phrase. The videos were clear and easy to follow. This meant volunteers required no prior BSL knowledge as they were given a tutorial on how to perform the selection of signs they were allocated. We narrowed down the collection to 22 such that the final group of phrases had overlapping words and sub-phrases to allow for the model to see the same words or phrases in different contexts.

However, because volunteers were only given a subset of the phrase set to perform and not all volunteers ended up supplying videos, after volunteering, the final data set was more abundant in certain phrases than others causing an imbalanced data set, shown in table 4.1

Phrase	Count
Can I Help You	13
Are You Worried	12
Are You Ok	11
Whats Your Name	11
Did You Wash Your Hands	11
Wheres The Pain	10
Are You Ill	10
Thank You	9
Wheres The Toilet	9
Finish Eating	9
Its Time To Sleep	9
Its Time For School	9
I Am Sad	9
Are You Happy Today	8
Are You Hungry	8
Its Sunny Today	8
Can I Play	8
I Am Happy	8
Are You Finished	7
How Are You Today	6
Im Sorry	6
Hello My Name Is	5

Table 4.1: *Final set of phrases with the number of videos for each phrase.*

Figure 4.2: *Example frames for the different movements that make up the phrase "it's time to sleep".*

When seeking volunteers they were assured that the videos would not be distributed and only be seen by the researchers. They were also informed that the videos would be deleted after submission and were only being used to obtain hand pose data.

4.1.2 Data Collection

I then passed each frame of the video through the MPH solution which identified landmarks for the hands in each frame, allocating each a coordinate position in the video space. Each video required the data for the coordinate positions to be normalised between 0 and 1. Now, the hand landmark data for a given video would be mapped to the corresponding phrase for the video. This meant for each video I now had a data mapping of the form;

$$\{[n_{frames}, n_{hands}, n_{landmarks}, n_{co-ords}] : phrase\} \tag{4.1}$$

Now with this data mapping, I was ready to begin the design of the ML model.

4.2 Model

The model was based on the Neural Translation model given in the Bahdanau et al. (2015) paper. This consisted of an encoder, attention layer and decoder.

This decision was made due to the problem of sign language translation being very similar to that of written language translation. Also, the conclusion of the Bahdanau et al. (2015) paper was that the NMT architecture saw far better translation for longer sentences than other state-of-the-art models at the time. So, with that, we will use the basic architecture of the NMT for this model. The NMT architecture like standard seq2seq uses an encoder and decoder but also adds an attention mechanism.

Some nice intuition behind how the attention layer works, making the model more effective on longer sentences, is if you were to imagine a professional interpreter attempting to translate a long speech they would not wait until the speech was finished to begin translating but rather translate words in smaller segments as they hear them. This is because they do not need to hear the entire speech to be able to translate a small segment. A small segment of the speech provides enough context for the interpreter to give an accurate translation for that segment (you 2018).

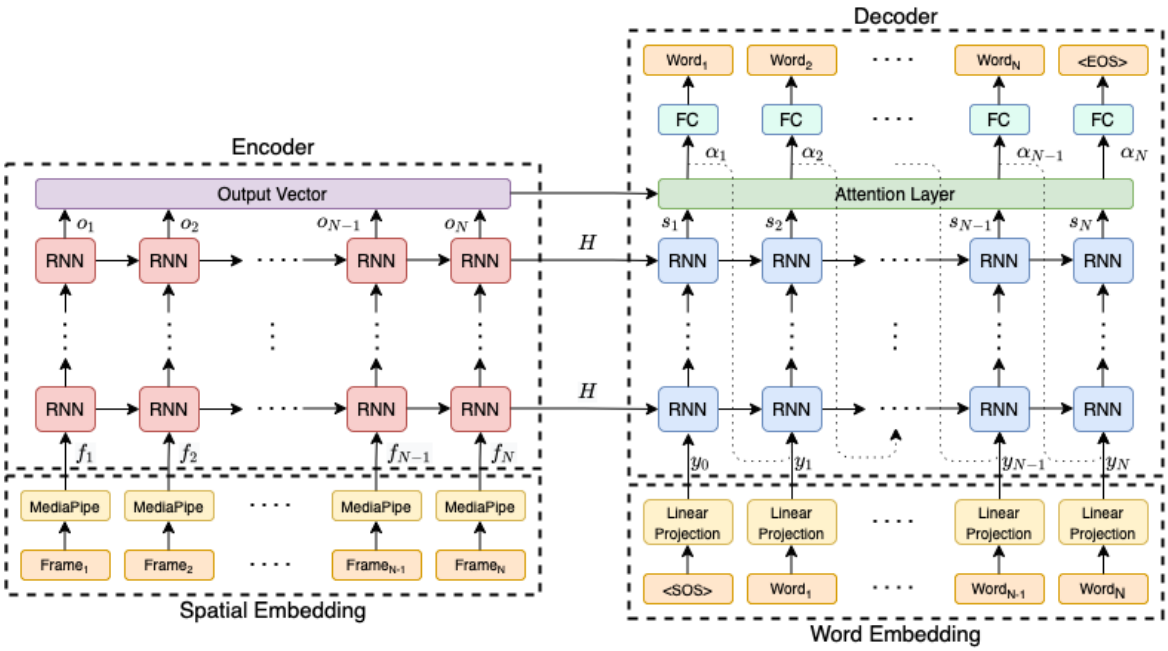


Figure 4.3: *Architecture overview of the method used to create written English outputs from a series of video frames, based on the diagram used in the Camgoz et al. paper Camgoz et al. (2018).*

4.2.1 Encoder

The encoder was designed using a multi-layered GRU, bi-directional recurrent neural network (BiRNN). The use of a BiRNN means that for each frame of the input data we want to consider all following frames but also all preceding frames. This should help when trying to unravel the difficult grammar that sign language poses for the task of translation. The use of more than one layer is an effort to improve the complexity of the model and hopefully amount to more accurate predictions, we will test this later on.

Spatial Embedding The spatial embedding encapsulates and extends all the pre-processing done to the data, explained in section 4.1. With the data in the form given in equation 4.1, we adjust it by flattening the data for each frame. So for each video, we have a tensor, V, which is then input to the encoder.

$$V_{shape} = [n_{frames}\ ,\ n_{hands} \times n_{landmarks} \times n_{co-ords}] \tag{4.2}$$

With this spatial embedding we now have the input for our encoder. The encoder is a function defined as;

$$O, H = Encoder(V) \tag{4.3}$$

where O is the output vector of the encoder such that $O = [o_1, o_2, ..., o_{N-1}, o_N]$ and H are the initial hidden states for the decoder

4.2.2 Attention

Now for the attention layer. This will take in the previous hidden state of the decoder, s_{t-1}, and all of the stacked forward and backward hidden states from the encoder, H, as inputs. The layer will output an attention vector, α_t, that is of length n_{frames}. Each element is between 0 and 1 and the entire vector sums to 1. The elements of α_t are the attention weights which say how much we should care about an input f_j when calculating an output word. This now means that our context vector, H, given by the encoder will become a weighted sum of the encoder outputs, weighted by the attention weights.

Intuitively, this attention layer takes what we have decoded so far, s_{t-1}, and all of what we have encoded, H, to produce a vector, α_t, that represents which frames in the source video we should pay the most attention to when predicting the next word.

4.2.3 Decoder

The decoder used is also a multi-layered GRU BiRNN with the same number of layers as the encoder. The decoder will take the output vector, O, from the encoder, the hidden states, H, and the actual word Y. The decoder is where we implement the attention layer to allow for the decoder the focus on the important parts of the output vector, O, when predicting the n^{th} word.

The decoder function is defined as;

$$P = Decoder(Y, O, H) \tag{4.4}$$

where P is the predicted phrase, O is the encoder output vector and H are the encoder hidden states and Y is the input vector such that $Y = [y_0, y_1, ..., y_{N-1}, y_N]$.

The decoder uses the attention layer which takes the previous hidden state, s_{t-1}, all of the encoder hidden states, G, and returns the attention vector, α_t.

5 Implementation

5.1 Data Collection and Representation

With the set of phrases defined we then had to obtain video footage of phrases being performed by volunteers. Each volunteer was given 5 phrases to perform along with video examples from the Guy's and St Thomas' NHS Foundation Trust's YouTube channel.

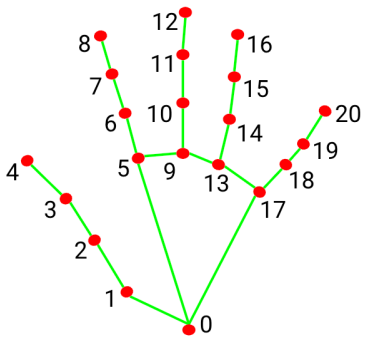
The phrases were allocated by giving volunteer 1, phrases 1 to 5 then volunteer 2, phrases 2 to 6 and so on. Once we reached phrase 22 we would just go back to phrase 1 and continue. For example volunteer 20 would sign phrases 20,21,22,1 and 2. Once the volunteers had completed their videos they would send them back to us. The volunteers who would record themselves performing the sign language phrases would do so without any intervention. They were asked to copy the videos to the best of their ability. This meant the videos we received were an attempt at copying someone else doing sign language.

Once the videos for each volunteer were received they were split into frames and passed through the MPH solution to obtain hand poses for both hands for each frame. The code we used to obtain hand pose using MPH was adapted from the MediaPipe GitHub page (med). An issue with MPH was that in some cases it would recognise the same hand twice. This resulted in a video of one person having pose data for more than 2 hands. We solved this issue because MPH labels a recognised hand as right or left. Using this, we would return hand pose data by taking the first instance of a right and left hand if they exist. If one hand did not exist then we would set all values to zero.

This solution allowed for an accurate depiction of a signer's left and right hand as 21 landmarks, respectively. These landmarks of the hand are shown in figure 5.1. Each landmark is a 3D point with an x, y and z coordinate. The z coordinate here represents how far away a given landmark is from the camera with the wrist being the zero point of the z-axis. For every video, v, in the set, we take every frame and pass it through the MPH solution to give us our 21 [x,y,z] points for each hand.

This would then result in obtaining data of size $[21, 3]$ for each hand [1] for each frame of the video. Thus results in a final data set of shape:

$$[n_{frames}, n_{hands} = 2, n_{landmarks} = 21, n_{co-ordinates} = 3] \tag{5.1}$$



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 5.1: *Specifies the landmarks obtained using the MPH solution. The numbers correspond to the indexing of the output. For example, in the list of landmarks, l, $l[8]$ would represent the tip of the index finger.*

[1] If part of a hand (or the entire hand) is not present in the frame then the corresponding landmarks will be set to zero.

The figures 5.2, 5.3 and 5.4 are to demonstrate the work done by the MPH solution. The 2D annotated images show its accuracy in picking out a hand from a 2D image. The 3D plots also show how landmark 'closeness' to the camera allows for a 3D model of the hand to be created. The 3D plots are examples of data we would have in our final data set for one hand in one frame.

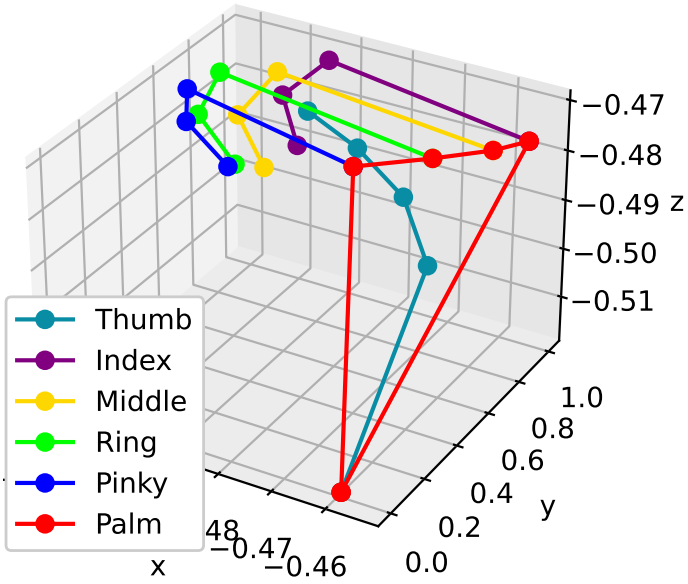


Figure 5.2: *An example 3D hand plot created from the pose of a knuckle obtained by the MediaPipe Hands solution. Note: The x and z axes are flipped to make the plot easier to interpret as a hand.*

Figure 5.3: *How the video data evolves into hand coordinate data. We start with the frame and pass it through MediaPipe which picks out the hand. We then have a 3D coordinate representation of the hand as shown in the plot. These 3D representations of the hand are what make up our data set.*

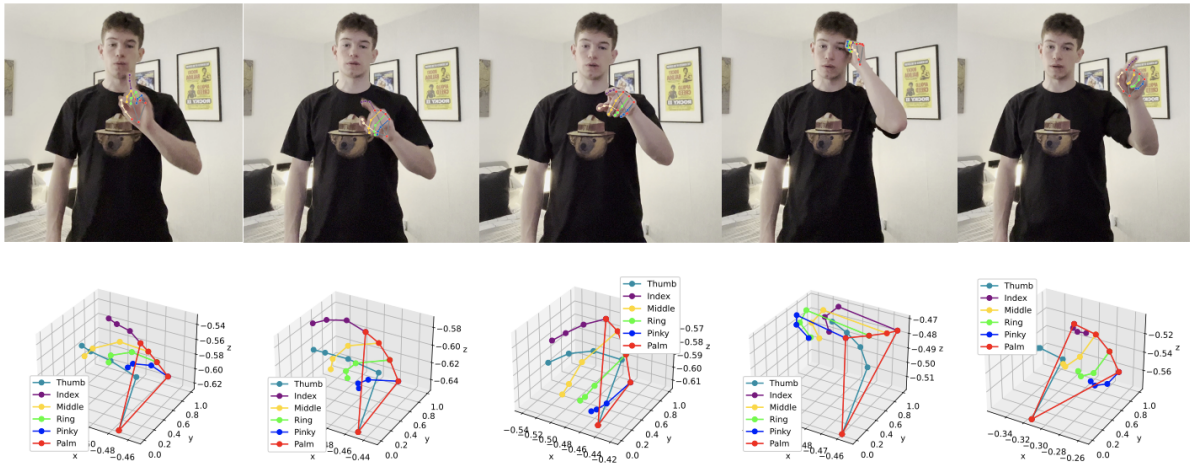


Figure 5.4: *Hand evolution throughout a video for the phrase "what's your name?", alongside the 3D hand plot.*

5.2 Data Augmentation

Data augmentation is the practice of making slight alterations to existing source data while mapping it to the same output to create new artificial data. Due to the time and resource constraints, the data set being used for the training of the model is scarce in the number of videos when compared with related pieces of work. To solve this issue I have made use of data augmentation with PyTorch when creating the data set.

The augmentations being performed on the data set are random re-scaling and random frame skipping. The re-scaling augmentation would multiply the x, y and z coordinates of the hand pose data for a given video by a random scaling factor respectively. The frame skipping augmentation we created would skip every m frames where m was a random integer and $m \leq n$.

The code for the random re-scaling and random frame skip augmentation are given below in 5.1 and 5.2

Listing 5.1: *Rescale a given sample input.*

```
class Rescale(object):
    """Rescale the hand plot in a sample by a given factor lambda,
       which will be a random float between 1 and 1-sf.
       where sf is the maximum scale percentage.

    Args:
        scale_percentage(int or float): Desired scaling factor
                                        for the data augmentation.
    """

    def __init__(self, scale_percentage):
        assert isinstance(scale_percentage, (int, float))
        self.sf = scale_percentage/100

    def __call__(self, sample):
        phrase = sample['phrase']
        landmarks = sample['landmarks']
        scaled_landmarks = np.copy(landmarks)

        x_y_z = [0,1,2]

        for ax in x_y_z:
```

```
            # set random variable which we use to scale
            # the landmarks in each of the x,y and z axes

            lambda = np.random.uniform(1-self.sf, 1)
            scaled_landmarks[:,:,:,ax] = landmarks[:,:,:,ax] * lambda

        return {'phrase': phrase, 'landmarks': scaled_landmarks}
```

Listing 5.2: Take every nth frame of a given sample input.

```
class SkipFrames(object):
    """Skip every n frames of an image where n
       is a random integer less than a declared
       maximum n.

    Args:
        max_n(int): The most frames to skip
                i.e. take every nth frame.
    """

    def __init__(self, max_n=1):
        assert isinstance(n, (int))
        self.max_n = max_n

    def __call__(self, sample):

        phrase = sample['phrase']
        landmarks = sample['landmarks']

        skip_n = random.randint(1,self.max_n)

        skipped_landmarks = landmarks[::skip_n]

        return {'phrase': phrase, 'landmarks': skipped_landmarks}
```

The goal for the re-scaling augmentation function was to mimic video data from people with different hand widths and heights as well as alter the proximity to the camera. The frame skip function was an attempt to imitate someone performing the signs at a faster speed. Another thought I had with the random re-scaling is that we could perhaps improve the performance of the model by reducing the length of the data while still keeping enough defining frames so that it may be easier for the model to recognise the main hand movements for a given sign language phrase. Since we augment the data after the use of the MPH solution we submit any opportunity to perform augmentation to the frames that could further improve the data set. These augmentation functions could be something like random image rotation, vertical and horizontal flipping or random cropping of an image.

5.3 Model

The implementation code was adapted from python notebooks found on the Bentrevett GitHub page.

5.3.1 Encoder

With the spatially embedded data as input, we initialise our encoder. We will be using a multi-layered GRU bi-directional RNN.

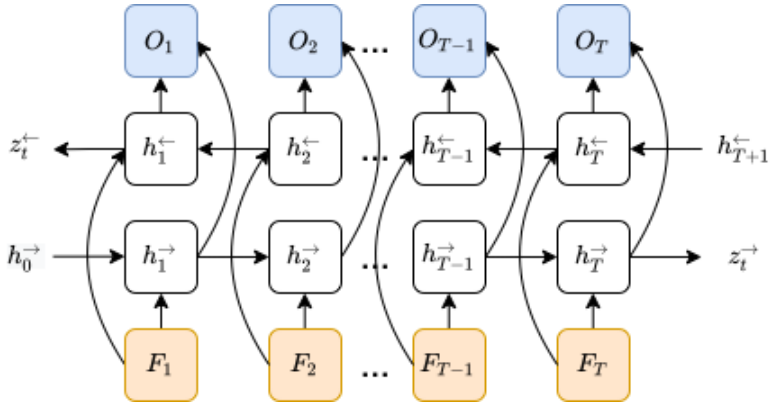


Figure 5.5: *Diagram illustrating the flow of inputs and outputs in a bi-directional RNN (BiRNN). Adapted from the diagram found here 9.4*

With the encoder architecture displayed in figure 5.2 each layer of the RNN will have two hidden layer outputs, $h_t^{\rightarrow}$ and $h_t^{\leftarrow}$, at time step t with;

$$\begin{aligned} h_t^{\rightarrow} &= EncoderGRU^{\rightarrow}(x_t\ ,\ h_{t-1}^{\rightarrow}) \\ h_t^{\leftarrow} &= EncoderGRU^{\leftarrow}(x_t\ ,\ h_{t+1}^{\leftarrow}) \end{aligned} \tag{5.2}$$

where $h_0^{\rightarrow}$ and $h_{T+1}^{\leftarrow}$ are pre-defined as initial hidden states which will be zero tensors. Also, F_t is the embedded input of frame t from a given sign language video.

The output vector will have shape: $[n_{frames}\ ,\ batch\_size\ ,\ hidden\_dimensions * n_{directions}]$. The last index having size $hidden\_dimensions \times 2$ is because for each RNN layer the output stacks the forward and backward hidden layers, $h_0^{\rightarrow}$ and $h_{T+1}^{\leftarrow}$, on top of each other. In other words we can represent the output vector as a variable O such that;

$$\begin{aligned} O &= [h_1, h_2, ..., h_T] \\ \text{given,}\quad h_t &= \begin{bmatrix} h_t^{\rightarrow} \\ h_t^{\leftarrow} \end{bmatrix} \end{aligned} \tag{5.3}$$

The hidden layer output from the RNN will be of size: $[n_{layers} * n_{directions}, batch\_size, hid\_dim]$. The first dimension means that for each GRU layer we have a forward RNN hidden layer with a backward RNN hidden layer stacked on top. However, because the decoder is not bidirectional, it requires a single context vector, z, to use as its initial hidden state, s_0, and we currently have two, $z^{\rightarrow} = h_T^{\rightarrow}$ and $z^{\leftarrow} = h_1^{\leftarrow}$.

To solve this we concatenate the two context vectors side by side with the forward layer preceding the backward layer. After we concatenate the hidden layers we pass them through a linear layer, g, and apply the tanh activation function. This leaves us with a final context vector z_l for GRU layer l such that;

$$z = \tanh(g(h_T^{\rightarrow}, h_0^{\leftarrow})) = \tanh(g(z^{\rightarrow}, z^{\leftarrow})) = s_0$$

Each GRU layer, l, will have a corresponding context vector z_l and because we are using a multi-layer decoder we can stack the context vectors, z_l, for each layer, l, and pass them through to the decoder as an initial hidden state, H.

$$H = \begin{bmatrix} z_L \\ z_{L-1} \\ \vdots \\ z_2 \\ z_1 \end{bmatrix} \tag{5.4}$$

where we have L GRU layers.

As we want our model to look back over the whole of the source sentence we return the output, O, as the stacked forward and backward hidden states for every frame in the source video. We also return the multi-layer context vector H which acts as our initial hidden state in the decoder.

5.3.2 Attention

Firstly, we calculate the *energy* between the previous decoder hidden state and the encoder hidden states. Since the encoder hidden states are a sequence of N tensors, and our previous decoder hidden state is a single tensor, the first thing we do is repeat the previous decoder hidden state N times. We then calculate the energy, E_n, between them by concatenating them together and passing them through a linear layer, lin, and a tanh activation function.

$$E_n = \tanh(\text{lin}(s_{n-1}, H))$$

where s_{n-1} is the decoder output for the previous word

We can think of this as evaluating how well each encoder's hidden state "matches" the previous decoder's hidden state.

To then obtain our attention we want to pass our energy through a linear layer, v, with the same length as the hidden dimension. This will result in an attention vector of the same length as the source sentence.

We can think of v as the weights for a weighted sum of the energy across all encoder hidden states. These weights tell us how much we should attend to each token in the source sequence. The parameters of v are initialized randomly but learned with the rest of the model through back-propagation. Note that v is not dependent on time, and the same v is used for each time step of the decoding.

$$\hat{\alpha}_t = v(E_t)$$

Finally, we ensure the attention vector fits the constraints of having all elements between 0 and 1 and the vector summing to 1 by passing it through a softmax layer.

We can now define our attention vector a_t as;

$$\alpha_t = \text{softmax}(\hat{\alpha}_t)$$

5.3.3 Decoder

The decoder uses the attention layer which takes the previous hidden state, s_{t-1}, all of the encoder hidden states, H, and returns the attention vector, α_t.

We then use this attention vector to create a weighted source vector, w_t, which is a weighted sum of the encoder hidden states, H, using α_t as the weights.

$$w_t = \alpha_t H \tag{5.5}$$

The embedded input word, y_t, the weighted source vector, w_t, and the previous decoder hidden state, s_{t-1}, are then all passed into the decoder's multi-layer GRU RNN, with y_t and w_t being concatenated together.

$$s_t = \text{DecoderGRU}(y_t, w_t, s_{t-1})$$

We then pass y_t, w_t and s_t through the linear layer, f, to make a prediction of the next word in the target sentence, $\hat{y}_{t+1}$. This is done by concatenating them all together.

$$\hat{y}_{t+1} = f(d(y_t), w_t, s_t)$$

6 Evaluation

The main objective of this project was to create an effective end to end BSL translation solution. When evaluating the models we use a pre-defined training and test set for all models to allow for a fair comparison. Also, due to limited time and resources, the models could only be trained for a limited amount of time with limited parameter settings. With that, it should be kept in mind that the final models here could be bested with more powerful computers.

6.1 Evaluation Metrics

When evaluating prospective models I chose to use BLEU (bilingual evaluation understudy) scores and accuracy.

BLEU is a method for evaluating machine-translated text by calculating the similarity of translations and the target phrase. The BLEU scores I use will be cumulative n-gram bleu scores, with $n \in 2, 3, 4$. This means that the BLEU-n scores will be an equally weighted sum of the BLEU scores for n-grams from 1 to n.

$$\text{BLEU-}n = \sum_{i}^{n} \frac{1}{n} BLEU_i \tag{6.1}$$

where $BLEU_i$ is the BLEU score comparing the similarity of the i-grams for two phrases.

The BLEU-n score will be between 0 and 1 where 0 means no n-grams in the prediction and target match and 1 means all n-grams in the prediction and target match.

Accuracy is the number of correct predictions over the total number of predictions made.

$$Accuracy = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{6.2}$$

The use of these metrics means that I can see how many predictions the model gets correct via accuracy and have a more forgiving metric of how close the predictions are via the BLEU scores.

6.2 Model Tuning

To evaluate our models we first need to tune their hyper-parameters. To tune the models I used a random search with 3-fold cross-validation. Each model was tuned on 10 hyper-parameters. The parameters and their possible values can be shown in table 6.1.

The random searches were completed using Weights&Biases, a helpful site that allows you to keep track of parameter configurations as well as log evaluation metrics for a given training run. (https://wandb.ai/site). I tuned the models on standard neural network parameters like hidden dimension, batch size and learning rate. I also tuned the models for *rescale percentage* and *frames to skip*. The *rescale percentage* value represented how big a possible rescaling of the hand in any sample could be. The *frames to skip* parameter represented the highest number of frames that would be skipped for a given sample.

Parameter	Min. Value	Max. Value	Type
Batch Size	1	64	Integer
Learning Rate	5e-5	5e-4	Float
Number of Layers	2	6	Integer
Decoder Dropout	0.4	0.9	Float
Encoder Dropout	0.4	0.9	Float
Hidden Layer Size	256	1024	Integer
Decoder Embedding Dimensions	256	1024	Integer
Rescale Percentage	1	14	Float
Frames to Skip	2	5	Integer

Table 6.1: *Value ranges for the hyper-parameters all models were tuned on*

6.3 Reduced Features vs Full Features

From initial testing, I found the model to be overfitting early in the training phase resulting in dummy predictors which would predict the same phrase for almost all input data. There could be several reasons for this overfitting. Overfitting can occur because of noisy data or the use of a training set that is too small as well as too complex a model. Biswal (2021). To try and overcome this problem I sought to try and reduce the complexity of the model. To do this I reduced the number of landmarks used from the input data which in turn reduced the number of embedding dimensions for the encoder (Recall, $Embedding\_Dimensioms = n_{hands} * N_{LANDMARKS} * n_{coords}$.

The reduction was done by considering only the palm and the endpoints of each finger. This can be shown in figure 6.1. I still consider the data in 3D however the plots are in 2D so the difference in representation is plainer to see.

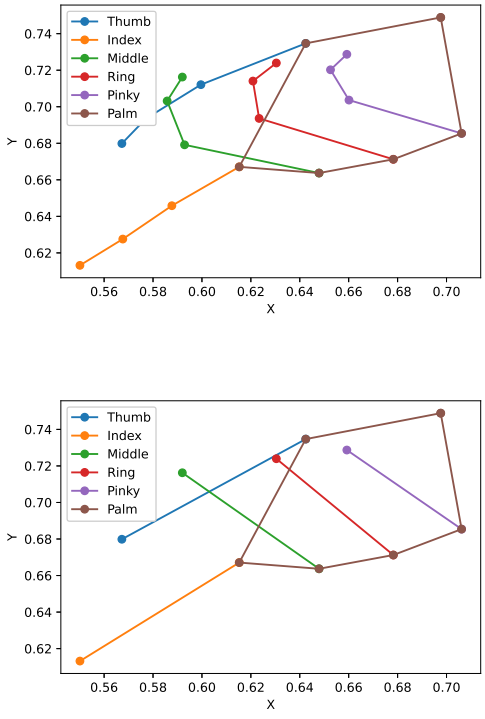


Figure 6.1: *Shows the difference in the full feature and reduced feature hand representations.*

6.3.1 Model Tuning

Figure 6.2 shows the results from the parameter tuning of the full and reduced feature models. Due to computational restrictions, I had to limit the number of layers to 6. For both models, the best loss is found using 6 layers. This allows for a fair comparison between the models since we have reason to believe both would perform better with an increased number of layers. The same argument would be true for the batch size and frames to skip parameters.

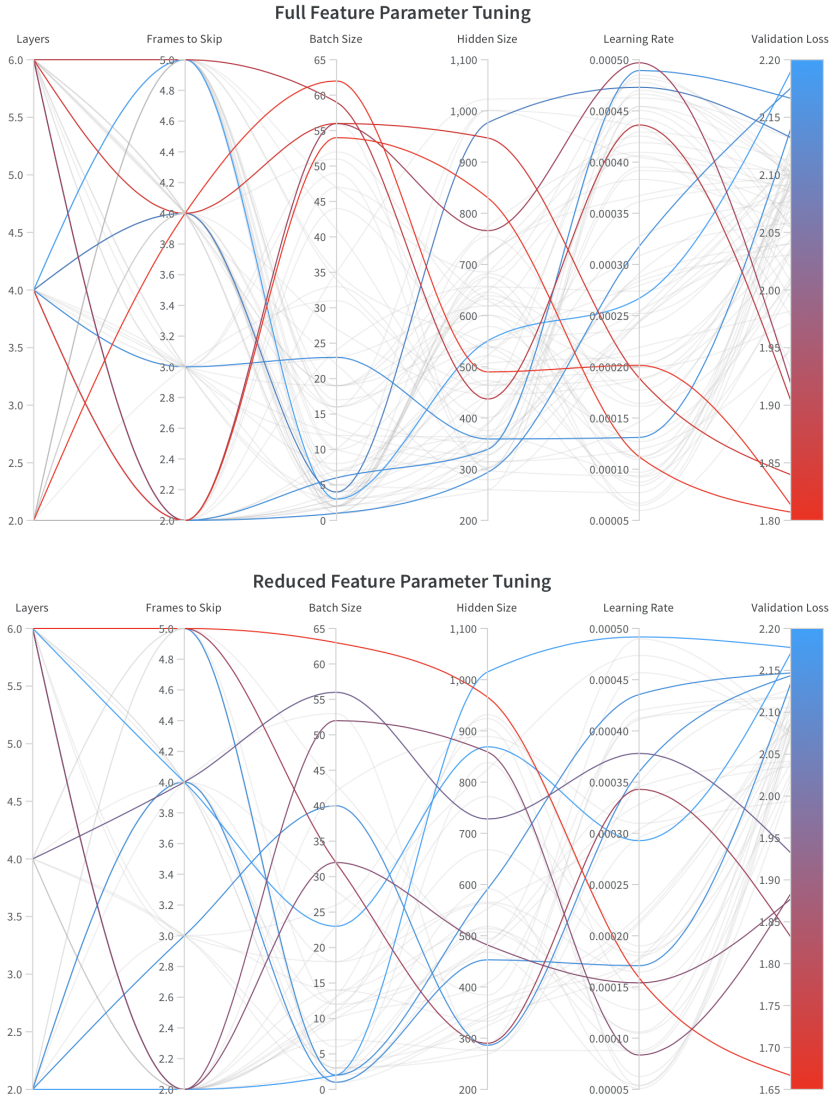


Figure 6.2: *Parallel coordinate plots for the most important hyper-parameters in training the models for the full feature and reduced feature data. The plots show the results from the parameter tuning displaying the parameter values that lead to the 5 best and 5 worst tuning runs. The plots show the 5 parameters that had the most influence on the validation loss. The most important parameters were found using the parameter importance feature on Weights&Biases.*

6.3.2 Results

After parameter tuning, I ran the best performing models for the respective data sets on a 70-15-15 train-validation-test split on the respective data sets. The results from the model training are shown below in figure 6.3. We can see from the figure that both models are subject to over-fitting. The full feature model finds a minimum validation loss of 1.77 and the reduced feature model finds a minimum validation loss of 1.73.

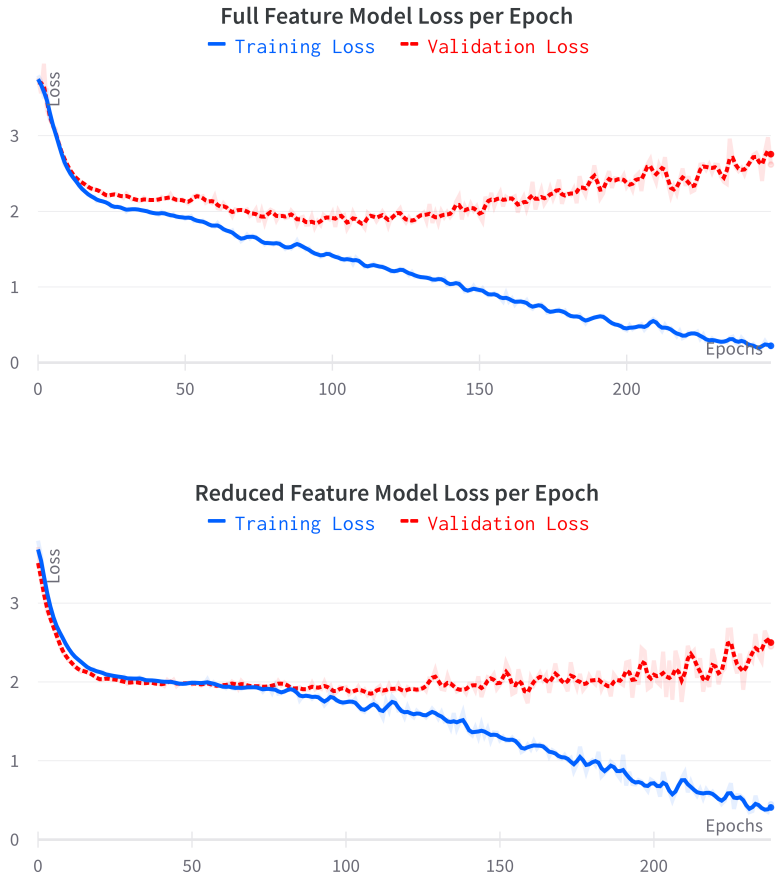


Figure 6.3: *Train and validation loss for the tuned model on the full and reduced feature data sets.*

The results of both models on the chosen evaluation metrics are shown in table 6.2

	Full Feature Set		**Reduced Feature Set**	
	Val Set	Test Set	Val Set	Test Set
BLEU-2	17.89	20.55	17.79	17.99
BLEU-3	11.19	15.09	11.24	10.32
BLEU-4	8.80	10.59	8.71	8.39
Accuracy	0	0.07	0	0

Table 6.2: *BLEU Scores and Accuracy for the Full and Reduced Feature Data Sets*

From the results in 6.2 we can see that in reducing the feature set there is no significant improvement or deterioration in the metric results. From this, I would conclude that the full feature set is not too complex. The 21 landmarks originally produced by the MPH solution then is a useful data set and required no feature reduction.

6.4 Unigram vs Bigram

Due to the topic comment nature and vocabulary of BSL, the correct prediction of a BSL phrase could include words that aren't being signed. For example, when someone performs the phrase "are you ok?" in BSL there is no point where they make a sign for the word "are". With this in mind, I consider increasing the number of words predicted at each time step by the decoder. So rather than the decoder outputting one word (or a

unigram) at each time step, I will test outputting two words (bigrams). Building a bigram model means that a different language is used for the output. In a unigram model, the language would be built up by all individual words that appear in the corpus of phrases. For example, the phrase "how are you today?" would be made up of language items "how", "are", "you" and "today". Whereas in a bigram model, where the language contains all ordered word pairs which appear in the phrases corpus, "how are you today?" would be made up of the items "how are", "are you" and "you today".

For the comparison, I will compare the previously trained full-feature model, which uses word unigrams by default, with a full-feature model using word bigrams. The bigram model will also be tuned using a 3-fold cross-validated random search similarly to the previously mentioned models.

The results of training the tuned bigram model are shown below in 6.4. From the results, we see that the bigram model is subject to the same over-fitting as the previously mentioned models but also the bigram model performs worse on the validation set with a minimum validation loss of 2.18.

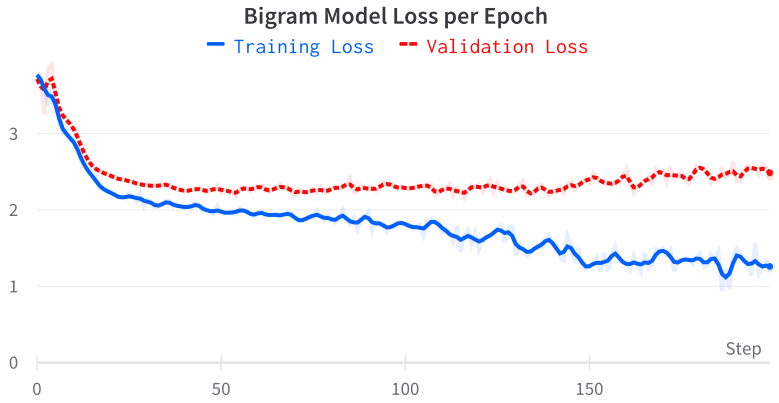


Figure 6.4: *Train and validation loss for the tuned model using bigrams*

6.4.1 Results

	Unigram		Bigram	
	Val Set	*Test Set*	*Val Set*	*Test Set*
BLEU-2	17.89	20.55	5.52	8.12
BLEU-3	11.19	15.09	4.36	6.71
BLEU-4	8.80	10.59	3.38	5.45
Accuracy	0	0.07	0	0

Table 6.3: *BLEU Scores and Accuracy for the Unigram and Bigram Models*

Target Phrase	Unigram Model	Bigram Model
im happy	are you hungry	can i you wash
did you wash your hands	wheres the toilet	are you
wheres the pain	are you worried	are you
thank you	are you hungry	are you
are you worried	are you name	are you
its time to sleep	finish eating happy	can i you wash
its time for school	wheres you toilet	are you
how are you today	are you toilet	are you i help
can i help you	are you hungry	are you
are you worried	whats your name	can i help
finish eating	are you	are you

are you ok	are you ok	are you
are you ok	are you toilet	are you i help
did you wash your hands	are you happy today	did you wash your
im happy	are you hungry	are you
its time to sleep	finish eating happy	can i help
are you ill	are you	are you
whats your name	whats your name	are you
are you worried	are you today	are you
can i play	are you worried	are you your name
how are you today	are you worried	are you
im sorry	are you hungry	are you
are you ok	are you name	are you
its time for school	are you	are you
did you wash your hands	are you hungry	are you
its sunny today	are you	are you
wheres the pain	are you name	whats your name
are you happy today	can i play	are you
thank you	finish eating	are you
are you ok	its time to sleep	can i help

Table 6.4: *Predictions for unigram and bigram models against the true phrase.*

From the results shown in 6.3 we can see the bigram model performs considerably worse on all metrics in comparison to the unigram model. From this I can conclude, with this data set, using word unigrams to build the language results in the most effective model.

6.5 Data

To evaluate the data set I will make points about the method of collection as well as the vocabulary and phrases used.

6.5.1 Data Collection

I would consider the method used for data collection, the MPH solution, to be very useful and effective for this problem. This is because the MPH solution was able to obtain hand pose data for all volunteers. This would ease data collection on a larger scale as volunteers can submit their video data at home using just a smartphone camera. This negates the need for volunteers to come to a meeting place to record their videos and means there is no requirement for long shoot days to obtain video data. The data set can be built dynamically as more videos are received from volunteers. Furthermore, the data collection method used mitigated the main privacy concerns from volunteers defined by Bragg et al. (2020) as no video data is required to build and train the model. Once a volunteer's video is passed through the MPH solution the videos can be deleted meaning volunteers need not worry about the videos being open to the public.

6.5.2 Phrase Set

Due to limited time and resources, the phrase set was built with the knowledge that it lacked size and diversity. I feel it is then unfair to criticise the data set in this regard.

The difference in phrase occurrences in the data set, previously shown in table 4.1, meant that the model will favour predicting certain sub-phrases as these sub-phrases appear more often in the training set so are more likely to be correct. For example, if you had a classification task and in your data set 90% of the examples were of class A and 10% were of class B then a dummy model that always predicted class A would have 90% accuracy. This is not an issue with the model but rather the product of an imbalanced data set.

If we look at the bigram occurrences for the phrases we see that the bigram "are you" is the most common in the data set by a large amount with 62 occurrences with the second most common being "can I" with 21 occurrences.

These occurrences can explain why we see the results that we do in table 6.4. In the bigram model the phrase "are you" is predicted far more often than anything else and could be classed as the dummy prediction. This common prediction of "are you" is the product of the imbalanced data set.

Table 6.5: *Data Set Phrase Counts*

Table 6.6: *Word Occurences*

Word	Count
You	95
Are	62
I	44
Its	26
Am	23
⋮	⋮
How	6
Sorry	6
Hello	5
My	5
Is	5

Table 6.7: *Bigram Occurences*

Bigram	Count
Are You	62
I Am	23
Can I	21
Wheres The	19
Its Time	18
⋮	⋮
You Today	6
Am Sorry	6
My Name	5
Name Is	5
Hello My	5

7 Conclusion

7.1 Project Summary

This project aimed to implement an end-to-end solution for translating BSL into English and seen the creation of a new BSL data set. The project succeeded in finding an end-to-end solution with a newly proposed method for data collection which saw the mitigation of privacy concerns from volunteers providing video data.

7.2 Future Work

7.2.1 Expanding the Data Set

In the future to improve the real-world effectiveness of the project, I would expand the data set. This would be done by incentivising volunteers to participate, via entry into a prize draw for example. Furthermore, due to the nature of the data collection, the volunteers could be from anywhere in the world (most likely extended to the whole of the UK considering the translation is for BSL).

7.2.2 Improve Model Complexity

In the future, with greater time and computational power, the model could be trained on more layers for instance with was unfeasible with the current computing power I had. This could then result in a better performing model than the one proposed in this project.

7.2.3 Evaluate Generalisation

To evaluate the generalisation of the model in further detail I would train the model using the same data set but the test set would be all the videos from one participant. This would allow me to see how the model performs when looking at videos from a signer it has never seen before. This was unfeasible for now as it would require training 38 different models (one for each test set).

7.3 Recommendation

The motivation for this problem was to create a solution that could help the Deaf and hard of hearing community feel less alienated from the general population. I feel that the SLT problem is solvable, given the creation of a large enough data set with diversity in vocabulary and signers. However, I feel that the simplest solution to making BSL users feel more welcome in society would be to make BSL a mandatory class for school children. Sign language is something children would pick up very easily. This would also solve the issue mentioned in section 2.1 that people who lose their hearing use English, as they most likely have no previous knowledge of BSL. In teaching BSL from a young age, it could mean that given someone loses their hearing, previous knowledge of BSL would allow them to fit into the Deaf community with more ease.

7 Bibliography

Learn to sign with evelina london. URL https://www.youtube.com/playlist?list=PLcaVWbSw-0R2D1kzIjIechpKQZMaAdrws.

Help amp; resources. URL https://bda.org.uk/help-resources/.

British sign language. URL https://bsl.surrey.ac.uk/principles/deaf-culture.

Hands. URL https://google.github.io/mediapipe/solutions/hands.html.

C5w3l07 attention model intuition, Feb 2018. URL https://www.youtube.com/watch?v=SysgYptB198.

History of sign language, Mar 2022. URL https://en.wikipedia.org/wiki/History_of_sign_language.

M. Aiken. An updated evaluation of google translate accuracy. *Studies in Linguistics and Literature*, 3(3), 2019. doi: 10.22158/sll.v3n3p253.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

Bentrevett. Bentrevett/pytorch-seq2seq: Tutorials on implementing a few sequence-to-sequence (seq2seq) models with pytorch and torchtext. URL https://github.com/bentrevett/pytorch-seq2seq.

A. Biswal. The complete guide on overfitting and underfitting in machine learning, Nov 2021. URL https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting.

D. Bragg, O. Koller, N. Caselli, and W. Thies. Exploring collection of sign language datasets: Privacy, participation, and model performance. *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, 2020. doi: 10.1145/3373625.3417024.

N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden. Neural sign language translation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/cvpr.2018.00812.

N. C. Camgöz, A. A. Kındıroğlu, and L. Akarun. Sign language recognition for assisting the deaf in hospitals. *Human Behavior Understanding*, page 89–101, 2016. doi: 10.1007/978-3-319-46843-3_6.

Linguisttraining.com. British sign language, Feb 2016. URL https://www.linguisttraining.com/news/2016/2/10/british-sign-language.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, 1988. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(88)90048-9. URL https://www.sciencedirect.com/science/article/pii/0031320388900489.

E. Vahdani, L. Jing, M. Huenerfauth, and Y. Tian. Recognizing american sign language manual signs from rgb-d videos. *SSRN Electronic Journal*, 2022. doi: 10.2139/ssrn.4019317.

H. Wang, X. Chai, X. Hong, G. Zhao, and X. Chen. Isolated sign language recognition with grassmann covariance matrices. *ACM Transactions on Accessible Computing*, 8(4):1–21, 2016. doi: 10.1145/2897735.